CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

The Generalized Bin Packing Problem

Mauro Maria Baldi
Teodor Gabriel Crainic
Guido Perboli
Roberto Tadei

May 2010

CIRRELT-2010-21

Bureaux de Montréal :
Université de Montréal
C.P. 6128, succ. Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

UNIVERSITÉ LAVAL · UQÀM · HEC MONTRÉAL · ÉCOLE POLYTECHNIQUE MONTRÉAL · Université de Montréal

The Generalized Bin Packing Problem

Mauro Maria Baldi[1], Teodor Gabriel Crainic[2,3*], Guido Perboli[1,2], Roberto Tadei[1]

[1] Department of Control and Computer Engineering, Politecnico di Torino, Corso Duca degli Abruzzi, 24 - I-10129 Torino, Italy

[2] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)

[3] Department of Management and Technology, Université du Québec à Montréal, P.O. Box 8888, Station Centre-Ville, Montréal, Canada H3C 3P8

Abstract. We introduce the Generalized Bin Packing Problem, a new packing problem where, given a set of items characterized by volume and profit and a set of bins with given volumes and costs, one aims to select the subsets of profitable items and appropriate bins to optimize an objective function combining the cost of using the bins and the profit yielded by loading the selected items. The Generalized Bin Packing Problem thus generalizes many other packing problems, including Bin Packing and Variable Sized Bin Packing, as well as Knapsack, Multiple Homogeneous and Heterogeneous Knapsack. We present two formulations for the problem, as well as an efficient column-generation-based lower bound method, plus first fit, best fit, and column-generation-based upper bound procedures. The paper introduces new instance sets and analyzes the results of extensive computational experiments, which show that the proposed procedures are very efficient and the bounds are very tight.

Keywords. Variable-size bin packing with fixed costs, heuristics, lower bounds, upper bounds.

Acknowledgements. Funding for this project has been provided by the Politecnico di Torino and the Natural Sciences and Engineering Research Council of Canada (NSERC), through its Industrial Research Chair and Discovery Grants programs, and by the partners of the Chair, CN, Rona, Alimentation Couche-Tard and the Ministry of Transportation of Quebec.

---

* Corresponding author: Teodor.Gabriel@cirrelt.ca

Dépôt légal – Bibliothèque nationale du Québec,
Bibliothèque nationale du Canada, 2010

1 Introduction

Packing problems consider sets of items and item-holding objects called bins, and aim to group items in such a way that they all (or a maximum number of them) fit into the minimum number of bins. The interest in packing problems follows from their own importance as a hard combinatorial optimization problem class (in all but their simplest form), as well as from the wide range of applications of theoretical and practical importance, e.g., loading, cutting, scheduling, and routing. Packing problems are particularly relevant for transportation and logistics, not only in the classical operational settings of loading items into "vehicles" (pallets, containers, trailers, trucks, rail cars, ships, and so on) and warehousing spaces, but also in numerous planning activities, from the design of plans and schedules to the participation to electronic markets.

The consideration of various rules and restrictions on the accommodation of items into bins, as well as of different item and bin attributes, has yielded several particular problem settings and a significant body of contributions (see (Wäscher et al., 2007) for a detailed survey). A number of issues and challenges remain, however. Thus, despite a number of recent efforts on the Variable Sized Bin Packing problem (Correia et al., 2008; Monaci, 2001; Crainic et al., 2010), there is still no general modeling framework able to address satisfactorily different variants of packing problems. The specification of objective functions particular to each problem variant appears as a major explaining factor to this situation (it is interesting to contrast this situation to the vehicle routing field where common structures in the objective function have yielded more general formulations and solution methods, e.g., (Cordeau et al., 2001; Pisinger and Ropke, 2007)). Moreover, there are no formulations simultaneously addressing the "cost" attributes of bins and items, problem setting of particular relevance for transportation and logistics. The objective of this paper is to contribute to address both these issues.

In this paper, we introduce a new packing problem, the *Generalized Bin Packing Problem* (*GBPP*) where, given a set of items characterized by volume and profit and a set of bins with given volumes and costs, one aims to select the subsets of *profitable* items and appropriate bins to optimize an objective function combining the cost of using the bins and the profit yielded by loading the selected items. The GBPP generalizes many packing problems present in the literature, Bin Packing and Variable Sized Bin Packing, as well as Knapsack, Multiple Homogeneous and Heterogeneous Knapsack, and so on. It also provides the means to identify and analyze the trade-off between item and bin selections that are part of many transportation and logistics planning problems.

We present two mixed integer programming formulations of the GBPP. The first is based on item-to-bin assignment decisions and requires a polynomial number of variables and constraints. This formulation is useful in discussing how the GBPP generalizes the packing problems mentioned above, but is not suitable for efficient computation. We thus introduce a second model based on feasible loading patterns and set covering

ideas. Despite requiring an exponential number of variables, this latter model facilitates the development of efficient solution methods. We thus present several procedures to compute lower and upper bounds for the GBPP and show their effectiveness through an extensive experimental phase.

The contributions of this paper are thus threefold. We introduce a new packing problem that is both relevant for many transportation and logistics planning problems and generalizes a wide range of packing and knapsack problems. We present two formulations for the problem, including one that yields an efficient column-generation-based lower bound method, as well as first fit, best fit, and column-generation-based upper bound procedures. An extensive set of experiments shows that the proposed procedures are very efficient and the bounds are very tight.

The paper is organized as follows. Section 2 recalls the main contributions to the Bin Packing and Variable Sized Bin Packing literature, the two problem settings the GBPP immediately generalizes. The formulations are introduced in Section 3, while lower and upper bounds are presented in Sections 4 and 5, respectively. The instance sets and extensive computational results are presented and discussed in Section 6. We conclude in Section 7.

2 Literature Review

The problem settings most immediately generalized by the GBPP are the Bin Packing and Variable Sized Bin Packing problems. We briefly recall the literature related to these problems.

The objective of the Bin Packing problem (BPP) is to load all the items while minimizing the number of used bins. The problem has been deeply studied in the past decades, producing several exact and heuristic methods (Martello and Toth, 1990). The first bounds were proposed by Martello and Toth (Martello and Toth, 1990). New lower bounds were developed some ten years later by Fekete and Schepers (Fekete and Schepers, 2001) by means of dual feasible functions. Starting from this paper, Crainic et al. (Crainic et al., 2007b,a) developed fast and more accurate lower bounds, able to reduce the optimality gap for a number of hand instances. A different approach was defined in (Vanderbeck, 1996), where the author proposes a formulation with an exponential number of variables and a column generation lower bound procedure for Bin Packing and Cutting Stock problems.

Several heuristics have also been proposed, e.g., the polynomial-time approximation schemes of e la VegaDE LA VEGA and Lueker (de la Vega and Lueker, 1981) and Karmarkar and Karp (Karmarkar and Karp, 1982) allowing to approximate an optimal

solution within $1+\epsilon$ for any fixed ϵ. However, these results are hardly usable in practice, due to the enormous size of the constants characterizing the polynomials. The literature thus shows that the most commonly used methods to achieve computational efficiency together with solution quality are the *First Fit Decreasing* (FFD) and *Best Fit Decreasing* (BFD) heuristics, sometimes combined with local improvement heuristics (Schwerin and Wäscher, 2006).

The Variable Sized Bin Packing problem (VSBPP) is a generalization of the BPP, where all the items must be loaded, but the bins can be chosen among several classes differing in volume and cost. The total accommodation cost, computed as the total cost of the used bins, must be minimized. A number of studies were recently dedicated to the VSBPP. Monaci in his Ph.D. thesis (Monaci, 2001) presented a series of lower bounds and solution methods (both heuristic and exact) for the VSBPP variant where the bins costs were set to their respective volumes. Kang and Park (Kang and Park, 2003) developed two greedy algorithms for another special case of the VSBPP, where the unit cost of each bin does not increase as the bin size increases, and analyzed their performances on instances with and without divisibility constraints (in the former case, on both the item and bin sizes, or on the bin sizes only). Seiden et al. (Seiden et al., 2003) proposed upper and lower bounds for the on-line version of the problem.

An integer linear formulation for the two-dimensional VSBPP was proposed by Pisinger and Sigurd (Pisinger and Sigurd, 2005), together with lower bounds obtained applying Dantzig-Wolfe decomposition and an exact branch-and-price algorithm. Alves and Valerio De Carvalho (Alves and Valerio De Carvalho, 2007) applied a column generation approach to the mono-dimensional problem and proposed a series of strategies aimed at accelerating the solution method. Correia et al. (Correia et al., 2008) proposed a formulation that explicitly included the bin volumes occupied by the corresponding packings, together with a series of valid inequalities improving the quality of the lower bounds obtained from the linear relaxation of the proposed model. The authors also introduced a large set of instances with up to 1000 items and used them to analyze the quality of the lower bounds. Finally, Crainic et al. (Crainic et al., 2010) considered the VSBPP with a general cost scheme for the bins, and proposed tight lower and upper bounds, which can be computed within a very limited computational time, and were able to solve to optimality all the instances proposed in (Correia et al., 2008).

3 Problem Definition and Formulations

The Generalized Bin Packing Problem considers a set of items characterized by volume and profit and sets of bins of various types characterized by volume and cost. Part of the items, which we denote *compulsory*, must be loaded, while a selection has to be made among the *non-compulsory* ones. The objective is to determine which non-

compulsory items to take and to select the bins to load the compulsory and the selected non-compulsory items to minimize the total cost, computed as the difference between the total cost of the used bins and the total profit of the loaded items.

We initiate this section with a formal description of the GBPP, followed by a mixed integer programming formulation based on item-to-bin assignments. We then discuss the relationships between the GBPP and several other packing problem settings. Finally, we introduce the set covering formulation of the problem.

3.1 Notation

Let $\mathcal{I}$ denote the set of items, with $n = |\mathcal{I}|$, and let w_i and p_i be the volume and profit of item $i \in \mathcal{I}$, respectively. Define $\mathcal{I}^{\mathrm{C}} \subseteq \mathcal{I}$ the subset of items that must absolutely be loaded, and $\mathcal{I}^{\mathrm{NC}} = \mathcal{I} \setminus \mathcal{I}^{\mathrm{C}}$ the subset of non-compulsory items some of which may be chosen if profitable. Let $\mathcal{J}$ denote the set of available bins, with $m = |\mathcal{J}|$, and assume each bin belongs to one of $\mathcal{T}$ types, the indicator function $\sigma(j)$ giving the type of any bin $j \in \mathcal{J}$. Define for each type $t \in \mathcal{T}$, the minimum U_t and maximum L_t number of bins of that type that may be selected, as well as the selection cost C_t and volume W_t of a bin of the class. To simplify the presentation and without loss of generality, we assume $W_t \geq w_i,\ \forall i \in \mathcal{I},\ \forall j \in \mathcal{J}$. Finally, denote $U \leq \sum_{t \in \mathcal{T}} U_t$ the maximum number of bins of all types one can use. The item-to-bin accommodation rules of the GBPP may then be stated as:

- All items in I^{C} must be loaded;
- For all used bins, the sum of the volumes of the items loaded into the bin must be less or equal to the volume of that bin;
- For all bin classes, the number of bins used for each class t must be within the lower and upper availability limits L_t and U_t;
- The total number of used bins cannot exceed U.

3.2 An assignment model of the GBPP

Consider the following decision variables:

- Bin selection binary variables y_j equal to one if bin $j \in \mathcal{J}$ is used, 0 otherwise;
- Item-to-bin assignment binary variables x_{ij} equal to 1 if item $i \in \mathcal{I}$ is selected (if non-compulsory) and loaded into bin $j \in \mathcal{J}$, 0 otherwise.

An assignment model for the GBPP can then be formulated as follows:

$$\text{Minimize} \quad \sum_{j \in \mathcal{J}} C_j y_j - \sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{I}^{\text{NC}}} p_i x_{ij} \tag{1}$$

$$\text{Subject to} \quad \sum_{i \in \mathcal{I}} x_{ij} w_i \leq W_j y_j \qquad j \in \mathcal{J}; \tag{2}$$

$$\sum_{j \in \mathcal{J}} x_{ij} = 1 \qquad i \in \mathcal{I}^{\text{C}}; \tag{3}$$

$$\sum_{j \in \mathcal{J}} x_{ij} \leq 1 \qquad i \in \mathcal{I}^{\text{NC}}; \tag{4}$$

$$\sum_{j \in \mathcal{J} | \sigma(j) = t} y_j \leq U_t \qquad t \in \mathcal{T}; \tag{5}$$

$$\sum_{j \in J | \sigma(j) = t} y_j \geq L_t \qquad t \in \mathcal{T}; \tag{6}$$

$$\sum_{j \in \mathcal{J}} y_j \leq U; \tag{7}$$

$$y_j \in \{0,1\}, \qquad j \in \mathcal{J} \tag{8}$$

$$x_{ij} \in \{0,1\}, \qquad i \in \mathcal{I}, \quad j \in \mathcal{J}. \tag{9}$$

The objective function (1) minimizes the total cost of the operations, where the total profit from the selected non-compulsory items is subtracted from the cost of using a number of containers of various types. Notice that the profit of the compulsory items is not included because it corresponds to a constant value the optimization cannot change. Regarding the type of optimization, we choose to present the minimization version to follow the tradition of packing problems. The equivalent formulation maximizing the total revenue (profit minus cost) recalls the knapsack problem settings.

Constraints (2) have the double effect of linking the usage of the bins and the accommodation of items into them and to limit the capacity of each used bin. Constraints (3) and (4) ensure that each compulsory and not-compulsory item is loaded into exactly and at most one bin, respectively. Constraints (5) and (6) enforce the maximum and minimum number of bins of each type one is allowed to use, respectively. Finally, constraint (7) limits the total number of bins selected to U.

We denote model (1)-(9) as *AM* and its continuous relaxation as *R-AM*. *AM* involves a polynomial number of variables and constraints. It is not well suited, however, to efficient algorithmic developments, due to the significant solution-space symmetry following from the item-to-bin assignment variables, which is typical of these compact models of packing problems. On the other hand, the model is suitable to observe the relationships between the problem we introduce and more classical packing problem settings.

3.3 Relationships to other packing problems

The GBPP generalizes several classical packing problems, including the Bin Packing problem (Martello and Toth, 1990), which may be obtained by considering a single bin type with $C_j = 1, j \in \mathcal{J}$ and $\mathcal{I}^{\mathrm{NC}} = \emptyset$, i.e., all the items must be loaded. Constraints (4) - (6) become then redundant and the objective becomes the minimization of the number of bins characteristic of the Bin Packing problem.

Allowing several bin types but still with $\mathcal{I}^{\mathrm{NC}} = \emptyset$ yields the Variable Sized Bin Packing problem where the total cost of the selected bins $\sum_{j \in \mathcal{J}} C_j y_j$ is minimized (constraint (4) is redundant) (Monaci, 2001; Crainic et al., 2010). Notice that an equivalent formulation of a Variable Sized Bin Packing problem can be obtained by imposing $\mathcal{I}^{\mathrm{C}} = \emptyset$ and setting the item profit higher than the cost of any bin class, $p_i > \max_{j \in \mathcal{J}} C_j$, which makes any container profitable even when only one item is accommodated into it.

The problem and formulation can similarly generalize several knapsack problems (see (Dyckhoff, 1990) for a detailed review). Since this underlines the generality of the problem we introduce in this paper, we give the main simplifications of the GBPP that yield each particular model:

- *Knapsack*: $|\mathcal{T}| = 1$, $|\mathcal{J}| = 1$, and $\mathcal{I}^{\mathrm{C}} = \emptyset$;
- *Homogeneous Multiple Knapsack*: $|\mathcal{T}| = 1$, $|\mathcal{J}| = m$, and $\mathcal{I}^{\mathrm{C}} = \emptyset$;
- *Heterogeneous Multiple Knapsack*: $|\mathcal{T}| > 1$, $|\mathcal{J}| = m$, and $\mathcal{I}^{\mathrm{C}} = \emptyset$.

3.4 Set Covering formulation of GBPP

We introduce in this subsection a set covering formulation for the GBPP based on feasible bin loading patterns.

A *feasible loading pattern* for a bin of type t corresponds to a set of items that may be loaded into the bin respecting all dimension restrictions and accommodation rules. Let $\mathcal{K}_t = \{k\}$ be the set of all feasible loading patterns for the bin type t, and $\mathcal{K} = \bigcup_{t \in \mathcal{T}} \mathcal{K}_t$. A feasible loading pattern k is described by a vector $\mathbf{A_k}$ of indicator functions a_k^i, $k \in \mathcal{K}_t$, $t \in \mathcal{T}$, such that $a_k^i = 1$ if item i is accommodated into pattern k of bin type t, 0 otherwise. The cost of pattern k is then the difference between the cost of the bin type and the profits of the non-compulsory items making up the pattern: $c_k = C_t - \sum_{i \in \mathcal{I}^{\mathrm{NC}}} a_k^i p_i$.

We define the bin loading pattern selection variables λ_k, equal to 1 if the pattern

$k \in \mathcal{K}_t$ is used, 0 otherwise. The set covering formulation of the GBPP may then be written as follows:

$$\text{Minimize} \quad \sum_{t\in\mathcal{T}}\sum_{k\in\mathcal{K}_t} c_k\lambda_k \tag{10}$$

$$\text{Subject to} \quad \sum_{t\in\mathcal{T}}\sum_{k\in\mathcal{K}_t} a_k^i\lambda_k = 1 \quad i \in \mathcal{I}^{\mathrm{C}}; \tag{11}$$

$$\sum_{t\in\mathcal{T}}\sum_{k\in\mathcal{K}_t} a_k^i\lambda_k \leq 1 \quad i \in \mathcal{I}^{\mathrm{NC}}; \tag{12}$$

$$\sum_{k\in\mathcal{K}_t} \lambda_k \leq U_t \quad t \in \mathcal{T}; \tag{13}$$

$$\sum_{k\in\mathcal{K}_t} \lambda_k \geq L_t \quad t \in \mathcal{T}; \tag{14}$$

$$\sum_{t\in\mathcal{T}}\sum_{k\in\mathcal{K}_t} \lambda_k \leq U; \tag{15}$$

$$\lambda_k \in \{0,1\} \quad k \in \mathcal{K}. \tag{16}$$

The objective function (10) minimizes the total cost of the selected bin loading patterns and, thus, of the selected bins and items. The feasibility of item-to-bin accommodations being guaranteed by the definition of the decision variables, constraints (2) are not required. Inequalities (11) and (12) have the same meaning as (3) and (4) in model AM expressing the need to load all compulsory items exactly once, while non-compulsory ones may be loaded at most once. Constraints (13), (14), and (15) replace (5) - (7) and enforce the bounds on the number of bins one is allowed to use for each type and globally, respectively.

We denote the formulation (10)-(16) as SC, and its continuous relaxation as $R\text{-}SC$. Compared to the assignment formulation, model SC has the advantage of separating the optimality and feasibility issues, the latter being addressed in the pattern generation, not directly considered in the model that focuses on identifying the optimal combination of patterns only. This makes SC easier to generalize to, for example, the two and three dimensional versions of the GBPP.

The separate consideration of the optimality and feasibility issues also makes SC a relaxation of AM. One then has $optimum(AM) \geq optimum(SC)$ and the lower bound obtained by continuous relaxation of AM is dominated by that provided by the linear relaxation of SC. Let $L_{R\text{-}AM} = optimum(R\text{-}AM)$ and $L_{R\text{-}SC} = optimum(R\text{-}SC)$. Then

Theorem 1. $L_{\text{R-AM}} \leq L_{\text{R-SC}}$.

Proof. We prove the theorem by contradiction. Let us suppose that it exists an instance

$\mathcal{I}'$ such that $L_{R\text{-}AM}(\mathcal{I}') > L_{R\text{-}SC}(\mathcal{I}')$. Let be $\overline{x_1}$ and $\overline{x_2}$ the optimal solutions associated to $L_{R\text{-}AM}(\mathcal{I}')$ and $L_{R\text{-}SC}(\mathcal{I}')$, respectively.

Given any solution x_2 of model *R-SC*, it is trivial to build an associated solution $x_1(x_2)$ of *R-AM* as follows:

- for any $\lambda_k > 0$, associate the pattern k to a different bin $\bar{j}_k$ of the proper type in model *R-AM* (recall that $k \in \mathcal{K}_t$, where t is a given bin type);
- given $\lambda_k > 0$ and the associated bin $\bar{j}_k$, put $x_{i\bar{j}_k} = a_k^i \lambda_k$.

Clearly, the objective functions of solutions x_2 and $x_1(x_2)$ have the same value by the definition of the pattern cost c_k. Moreover, all the constraints of *R-AM* are satisfied because λ_k is a feasible bin loading pattern. Let $\overline{x_2}$ be the optimal solution associated to $L_{R\text{-}SC}(\mathcal{I}')$. Then, by the previously described rule, we can build a feasible solution $x_1'(\overline{x_2})$ of *R-AM* with objective function $L_{R\text{-}SC}(\mathcal{I}') < L_{R\text{-}AM}(\mathcal{I}')$. This implies that $\overline{x_1}$ is not optimal, which is in contradiction with the hypotheses. □

The set covering formulation requires a potentially exponential number of variables, however. We thus use column generation to compute a lower bound for the GBPP.

4 Lower Bound through Column Generation

The lower bound is computed from the linear relaxation of the set covering formulation through column generation, which provides the means to implicitly deal with the large number of variables in the model (and is widely used in packing problems (Alves and Valerio De Carvalho, 2007; Vanderbeck, 1996)).

The general column-generation approach applied to our problem consists in starting with a relatively small set of feasible patterns $\mathcal{P}$ corresponding to a restricted GBPP that we denote R-GBPP. One then first solves the linear relaxation of the set covering formulation *SC* of R-GBPP, and then attempts to generate new feasible patterns with negative reduced cost. If successful, these are added to $\mathcal{P}$ and the procedure is restarted; otherwise, one has identified the optimal solution to *R-SC* and the procedure stops with a lower bound for GBPP. Formally,

1. Find an initial feasible solution to GBPP and the corresponding set $\mathcal{P}$;
2. Solve the linear relaxation *R-SC* of R-GBPP and let $L_{\text{R-GBPP}}$ be its optimal solution;

3. For each container type $t \in \mathcal{T}$
   (a) Find the non-basic pattern variable $\lambda_{\overline{k}}$ of $L_{\text{R-GBPP}}$, $\overline{k} \in \mathcal{K}_t$, with the smallest reduced cost $r_{\overline{k}}$ among all non-basic pattern variables for type t;
   (b) If $r_{\overline{k}} < 0$, $\mathcal{P} = \mathcal{P} \cup \{\lambda_{\overline{k}}\}$;
   (c) Continue to the next bin type (go to 3a);
4. If $r_{\overline{k}} \geq 0$ for all bin types t, then stop with $L_{\text{R-GBPP}}$ as the set covering lower bound for GBPP; Otherwise, go to 2.

The main issue is how to find new negative reduced cost feasible load patterns. Consider the dual variables associated to the constraints of the continuous relaxation of the R-GBPP set covering formulation:

- $\mu_i \geq 0$ for constraints (11);
- $\nu_i \leq 0$ for constraints (12);
- $\alpha_t \leq 0$ for constraints (13);
- $\beta_t \geq 0$ for constraints (14);
- $\epsilon \leq 0$ for constraint (15).

The reduced cost of a given pattern variable k for a bin of type t, expressed as $c_k - [\mu\ \nu\ \alpha\ \beta\ \epsilon]^T\ \mathbf{A_k}$, then becomes

$$\begin{aligned} r_k &= C_t - \sum_{i \in \mathcal{I}^{\text{NC}}} p_i\, a_k^i - \left[\mu^{\mathbf{T}}\ \nu^{\mathbf{T}}\ \alpha^{\mathbf{T}}\ \beta^{\mathbf{T}}\ \epsilon\right] \mathbf{A^k} \\ &= C_t - \sum_{i \in \mathcal{I}^{\text{NC}}} p_i\, a_k^i - \sum_{i \in \mathcal{I}^{\text{C}}} \mu_i\, a_k^i \\ &\quad - \sum_{i \in \mathcal{I}^{\text{NC}}} \nu_i\, a_k^i - \sum_{t \in \mathcal{T}} \alpha_t - \sum_{t \in \mathcal{T}} \beta_t - \epsilon \\ &= C_t - \sum_{i \in \mathcal{I}^{\text{NC}}} (p_i + \nu_i)\, a_k^i - \sum_{i \in \mathcal{I}^{\text{C}}} \mu_i\, a_k^i \qquad (17) \\ &\quad - \sum_{t \in \mathcal{T}} (\alpha_t + \beta_t) - \epsilon. \end{aligned}$$

We now define a column generation subproblem which, given a bin of type t, finds the non-basic pattern with the minimum reduced cost. Notice that the vector $\mathbf{A_k}$ defining a not yet generated pattern k of bin type t is not known but may be expressed in terms of

item-to-bin assignment variables x_i equal to 1 if item i belongs to the pattern, 0 otherwise (for simplicity of notation, we dropped the k index). Because the dual variables α_t, β_t, and ϵ, as well as the bin cost C_t, are constant for a given type t, the reduced cost of the yet-unknown pattern one desires to minimize becomes

$$\begin{aligned} r_k &= \left\{ C_t - \sum_{i \in \mathcal{I}^{NC}} (p_i + \nu_i)\, x_i - \sum_{i \in \mathcal{I}^{C}} \mu_i\, x_i \right. \\ &\quad \left. - \sum_{t \in \mathcal{T}} (\alpha_t + \beta_t) - \epsilon \right\} \\ &= \left\{ - \sum_{i \in \mathcal{I}^{NC}} (p_i + \nu_i)\, x_i - \sum_{i \in \mathcal{I}^{C}} \mu_i\, x_i \right\}. \end{aligned} \tag{18}$$

Finding the feasible pattern with minimum reduced cost for bin type t then becomes a knapsack problem:

$$\begin{aligned} \text{Maximize} \quad & \left\{ \sum_{i \in \mathcal{I}^{NC}} (p_i + \nu_i)\, x_i + \sum_{i \in \mathcal{I}^{C}} \mu_i\, x_i \right\} \\ \text{Subject to:} \quad & \sum_{i \in \mathcal{I}} w_i x_i \leq W_t \qquad t \in \mathcal{T} \\ & x_i \in \{0,\ 1\} \qquad i \in \mathcal{I}. \end{aligned}$$

Any feasible solution may be used to initialize the procedure, including the trivial solution obtained loading each compulsory item into a different bin. More accurate heuristics are presented in Section 5. The procedure adds at most $|\mathcal{T}|$ columns to $\mathcal{P}$ at every iteration (in Step 3), one for each bin type yielding a feasible loading pattern with r_k negative.

5 Upper bounds

In this section we present heuristic-based upper bounds for the GBPP. The first two are constructive heuristics extending ideas that have proved of great interest for Bin Packing and Knapsack problems. The third heuristic starts from the set covering lower bound, that is, from the optimal solution obtained at the end of the column generation procedure (when not integer).

5.1 Constructive heuristics

We propose heuristics to load items into bins derived from the well-known *First Fit Decreasing* (FFD) and *Best Fit Decreasing* (BFD) heuristics for the Bin Packing problem. Briefly, following an initial sorting of the items in non-increasing order of their volumes, the FFD loads items one after the other into the first bin where they fit, while BFD attempts to load each item into the "best" bin where it can be accommodated, usually the bin which, after loading the item, has the minimum *free volume*, defined as the bin volume minus the sum of the volumes of the items it contains. Both heuristics create a new bin when an item cannot be accommodated into an existing bin. Despite their simplicity, the FFD and BFD heuristics offer good performances for the Bin Packing problem (Martello and Toth, 1990).

Applying the first or best fit idea to the GBPP presents a number of challenges, however, which follow from the heterogeneity of the bins and the interplay among bin cost, non-compulsory item profit, and the volumes of the respective bins and items. First, one has to sort not only items but bins as well. Furthermore, while volumes are of course important, costs and profits could influence the order as well. Several strategies are thus possible and we examine four, which have all in common to place the compulsory items first in non-increasing values of their volumes. The four strategies for sorting the bins and the non-compulsory items (placed after the compulsory ones in the list) are

1. **Bins:** Non-decreasing order of C_j/W_j and non-decreasing values of their volumes;
   Non-compulsory items: Non-increasing p_i/w_i and non-increasing values of their volumes;
2. **Bins:** Non-decreasing order of C_j/W_j and non-decreasing values of their volumes;
   Non-compulsory items: Non-increasing volumes w_i and non-increasing values of p_i/w_i;
3. **Bins:** Non-decreasing order of C_j/W_j and non-increasing values of their volumes;
   Non-compulsory items: Non-increasing p_i/w_i and non-increasing values of their volumes;
4. **Bins:** Non-decreasing order of C_j/W_j and non-increasing values of their volumes;
   Non-compulsory items: Non-increasing volumes w_i and non-increasing values of p_i/w_i.

Given ordered sets of bins and items, the FFD and BFD heuristics proceed according to the standard sequence, the former loading each item into the first already-selected bin with sufficient available volume, while the latter selects the existing bin maximizing the MERIT FUNCTION$(i, b) = \frac{1}{r_b+1}$, where r_b stands for the resulting bin free volume as defined

above. In both cases, the next bin in the list is selected when a new bin is required to load a compulsory item.

Algorithm 1 The PROFITABLE(item i, container b) Heuristic for New Bin Selection

$\mathcal{I}^*$: sublist of $\mathcal{I}$ starting from item i;
Load i into b and initialize the container profit $P_b = p_i$;
for all $i' \in \mathcal{I}^*$ **do**
  if i' can be loaded into b **then**
    Load i' in b and update the container profit $P_b = P_b + p_{i'}$;
  end if
  if $P_b > c_b$, return TRUE **else** return FALSE.
end for

Major differences with the FFD and BFD for the BPP are that 1) one must chose a profitable subset of non-compulsory items to load, that is, items for which their total profits decrease as much as possible the total cost of all the selected bins, and 2) one must choose the bin to add to the solution given its particular cost and volume. Consider a non-compulsory item i that cannot be loaded into any partially loaded container and a new one should be selected. When the profit of i and those of the remaining items in the list are so low that their sum does not exceed the cost of the new container, then it is better not to select item i. To determine whether this is the case, it is sufficient to solve a knapsack problem considering the not yet loaded items and each bin type such that at least one bin is still available. To avoid the extra computational burden, which could be significant, we take advantage of the ordering of the bins and use the Algorithm 1 to compute an approximation of the value of selecting bin b for item i.

Another issue one needs to address is the behavior of the procedures toward the end of the item list, when there might not be sufficient items left to fill up the selected bin, even though it offers a good cost/volume ratio measure. A bin with a worst ratio but an absolute smaller cost might be appropriate in this case, and we include a post-processing phase that attempts to improve the solution by evaluating such possible bin swaps. The procedure iteratively examines each selected bin $j \in \mathcal{S}$ with a loaded volume U_j defined as the sum of the volumes of the items assigned to it. Then, if an unused bin $k \in \mathcal{J}$ exists such that $W_k \geq U_j$ and $C_k < C_j$, the items from bin j are transferred to bin k and bin j is discarded. Algorithm 2 displays the FFD and BFD heuristics developed for the GBPP.

5.2 Column Generation-based Diving Heuristics

Two approaches for computing upper bounds starting from the results of the column generation-based solution to the relaxation *R-SC* of the set covering formulation *SC*.

Algorithm 2 The BFD and FFD Upper Bound Heuristics

Sort the bins and the items (compulsory items first), and let SCL and SIL be the resulting ordered lists, respectively.
Let $\mathcal{S}$ be the set of the selected containers.
for all $i \in SIL$ **do**
  Identify the bin $b \in \mathcal{S}$ into which item i can be loaded:

- BFD: the best, i.e., the bin b maximizing the MERIT FUNCTION(i, b);
- FFD: the first with sufficient empty space to accommodate i.

  if b does not exist **then**
    if $i \in I^C$ **then**
      Identify the first container $b' \in SCL \setminus \mathcal{S}$ such that $w_i \leq W_{b'}$.
    else
      Identify the container $b' \in SCL \setminus \mathcal{S}$ such that PROFITABLE(i, b') returns TRUE.
    end if
    $b = b'$, if b' exists.
  end if
  If b exists, load i into b, reject item i, otherwise.
end for
for all $j \in \mathcal{S}$ **do**
  for all $k \in \mathcal{J} \setminus \mathcal{S}$ **do**
    $U_j = \sum_{i \text{ loaded into } j} w_i$
    if $W_k \geq U_j$ and $C_k < C_j$ **then**
      Move all the items from j to k
      $\mathcal{S} = \mathcal{S} \setminus \{j\} \cup \{k\}$
    end if
  end for
end for

Algorithm 3 The DIVING Heuristic

Let $\overline{x}$ be the optimal solution of the continuous relaxation of R-GBPP
while $\overline{x}$ is not integral **do**
  Select a non-integer variable λ_k and set $\lambda_k = 1$;
  Re-optimize R-GBPP.
end while

The first approach is to solve exactly the formulation MIP SC, by branch-&-bound, for example, considering only the columns generated by the column generation procedure while computing the lower bound. This may still be quite time consuming, however. Consequently, we stop the branch-&-bound after a given computation time and denote Z_{SC} the resulting value of the objective function and upper bound.

The second approach is based on *diving*, a well-known method for finding good quality integer solutions from continuous solutions to the respective relaxations (Atamturk and Savelsbergh, 2005). The working principle is to iteratively round up variables and re-optimize the continuous relaxation. Adapted to the GBPP, the diving heuristic mechanism is displayed in Algorithm 3.

The heuristic assumes that the optimal bin loading patterns of the GBPP are in the restricted set corresponding to the R-GBPP and slightly and iteratively perturbs the optimal continuous solution by fixing patterns to push toward an integer solution. The key feature is how to choose the variable to be fixed. Preliminary experiments have shown that criteria based on the contributions of the items making up the pattern to its marginal cost yield superior performances. Two criteria, selecting the pattern variable with non-integral value maximizing expression (19) or (20), make up the two diving heuristics included in the final comparative experiments of Section 6.

$$\sum_{i \in \mathcal{I}^{NC}} \nu_i \, a_k^i + \sum_{i \in \mathcal{I}^{C}} \mu_i \, a_k^i; \tag{19}$$

$$(1 - \lambda_k) \left(\sum_{i \in \mathcal{I}^{NC}} \nu_i \, a_k^i + \sum_{i \in \mathcal{I}^{C}} \mu_i \, a_k^i \right). \tag{20}$$

6 Computational results

The goal of the numerical experiments was to explore the behaviour and performance of the proposed heuristics and, thus, to better understand the GBPP and how to address this new class of packing problems.

The algorithms were coded in XPress IVE, while the instances of model SC needed by the column generation procedure, as well as the Z_{SC} and diving upper bound heuristics were solved by the XPress 2008a solver (Dash Associates, 2008). Experiments were conducted on a Pentium IV 3.0 Ghz workstation with 2 Gb of RAM.

No instances are present in the literature for the GBPP. We generated 900 new instances, partially based on those for the Variable Sized Bin Packing and the Bin Packing problems (Monaci, 2001; Vanderbeck, 1996; Crainic et al., 2010; Correia et al., 2008). Ten instances were randomly generated for each combination of number of items, item profit

and volume, and bin types defined as follows:

- Number of items: 25, 50, 100, 200, and 500.
- Item profit:

Class 0: All the items are compulsory;

Class 1: $p_i \in \lceil U(0.5, 3)w_i \rceil$, where U stands for the uniform distribution; No compulsory items present;

Class 2: $p_i \in \lceil U(0.5, 4)w_i \rceil$ and no compulsory items;

- Items volume:

  I1: [1, 100];

  I2: [20, 100];

  I3: [50, 100].

- Number of bin types: The cost of each bin is equal to its volume for

  A Three types with volumes respectively of 100, 120 and 150.

  B Five types with volumes respectively of 60, 80, 100, 120 and 150.

We first examine the general performance of the proposed heuristics. We then examine more in detail the impact of the relative distribution of the numbers of compulsory and non-compulsory items, as well as that of the number of items, on this performances.

6.1 General performance results

We first examine the behavior of the column generation-based lower bounds as illustrated in Tables 1 and 2. The former reports the results on the instances with item profit of Class 0. These instances are equivalent to instances for the Variable Sized Bin Packing (compulsory items only) and, thus, optimal solutions are known for most (Monaci, 2001; Crainic et al., 2010; Correia et al., 2008), allowing a finer analysis. Columns 1 and 2 report the numbers of bin types and items, respectively. Columns 3, 4, and 5 display the mean objective-function value, the average optimality gap (in %), and the number of proven optima for *R-SC*, the column generation-based lower bound, while Columns 6, 7, and 8 report the same information for the upper bound Z_{SC} obtained after 20 CPU seconds of solving the MIP formulation *SC* considering only the columns generated while computing the lower bound. Averages are computed over the 30 instances that correspond to each combination of bin types and numbers of items.

BIN TYPES	ITEMS	*R-SC*			Z_{SC}		
		VALUE	GAP	OPT	VALUE	GAP	OPT
3	25	1607.54	0.27	13	1611.67	0.03	29
	50	3133.71	0.18	5	3141.67	0.09	27
	100	6257.76	0.06	8	6266.00	0.08	20
	200	12469.18	0.04	6	12488.33	0.13	17
	500	31303.53	0.02	3	31379.67	0.22	15
5	25	1591.99	0.21	13	1595.00	0.00	30
	50	3154.74	0.12	9	3158.33	0.00	30
	100	6373.85	0.07	6	6378.67	0.01	28
	200	12463.53	0.04	6	12468.67	0.01	27
	500	31381.67	0.01	6	31390.33	0.02	21

Table 1: Column-generation lower bounds for item-profit class 0

The first thing one notices examining the results of Table 1 is that the column-generation procedure yields tight lower bounds, with a mean of about 0.1% and 75 instances solved to optimality. Moreover, the set of patterns it generates includes most of the variables making up the optimal solution to the original set covering formulation *SC*. Indeed, applying a commercial branch & bound to the *SC* MIP using only this set of patterns provides solutions with an average gap of less than 0.2% in 20 CPU seconds (gap that decreases to 0.05% when 30 seconds are allowed, thus addressing the 3-bin and 500-item case) and solves most instances, 244 over 300, to optimality.

Table 2 reports the overall results for all combinations of item profit class, number of bin types, and number of items, as indicated in Columns 1, 2 and 3, respectively. Columns 4 and 5 display the mean objective-function value for the column generation-based lower bound and the mean number of corresponding patterns that were generated. Column 6 gives the mean objective function of Z_{SC} for 20 CPU seconds, while Columns 7 and 8 display the mean "optimality" gaps (in %) between these lower and upper bounds, and the number of instances proven to be optimal (equal values for the two bounds), respectively. Notice that, the column generation-based lower bound and the mean objective function of Z_{SC} assume negative values for the instances of Classes 1 and 2. This follows from the the fact that all items in these instances are non compulsory and, thus, their profits are accounted for in the total cost, which is minimized, and are greater than the cost of the bins they are loaded into. The negative values thus indicate a revenue corresponding to the selected items and bins. Class 0 instances have all items compulsory (with no explicit profits) and thus the positive values displayed correspond to the total cost of the selected bins only.

The results confirm the excellent performance of the column generation-based lower bound procedure, which yields very low optimality gaps for all classes of instances and

CLASS	BINS	ITEMS	*R-SC*	PATTERNS	Z_{SC}	GAP	OPT
0	3	25	1607.54	110.10	1611.67	0.26	12
		50	3133.71	246.87	3141.67	0.25	3
		100	6257.76	519.27	6266.00	0.13	6
		200	12469.18	1065.57	12488.33	0.15	3
		500	31303.53	2808.10	31379.67	0.27	2
	5	25	1591.99	130.23	1595.00	0.19	13
		50	3154.74	282.50	3158.33	0.11	9
		100	6373.85	571.93	6378.67	0.08	4
		200	12463.53	1172.07	12468.67	0.04	4
		500	31381.67	3112.00	31390.33	0.03	2
MEAN						**0.12**	
1	3	25	-1260.49	56.57	-1257.77	0.22	15
		50	-2449.51	128.13	-2444.27	0.21	4
		100	-4931.44	247.03	-4923.43	0.16	2
		200	-9932.75	505.83	-9924.57	0.08	3
		500	-25268.70	1361.33	-25235.97	0.13	1
	5	25	-1269.84	65.30	-1267.07	0.22	13
		50	-2483.14	145.50	-2480.70	0.10	12
		100	-5196.42	289.13	-5194.53	0.04	10
		200	-10173.20	561.27	-10170.97	0.02	6
		500	-25601.17	1461.57	-25595.13	0.02	4
MEAN						**0.13**	
2	3	25	-2117.11	64.03	-2113.23	0.18	11
		50	-3868.00	140.73	-3862.83	0.13	4
		100	-8024.54	284.73	-8019.83	0.06	7
		200	-16185.98	554.77	-16176.03	0.06	3
		500	-40206.50	1391.80	-40155.10	0.13	1
	5	25	-1894.28	74.17	-1892.07	0.12	15
		50	-4089.92	162.20	-4087.20	0.07	7
		100	-8067.18	313.67	-8064.23	0.04	6
		200	-16019.83	618.83	-16015.87	0.02	4
		500	-40166.43	1616.60	-40158.00	0.02	4
MEAN						**0.09**	
OVERALL MEAN						**0.11**	

Table 2: Column-generation lower bounds for all instances

even identifies a good number of optimal solutions. The results also confirm that the set of generated patterns includes most of those making up the optimal solution, the average optimality gap being of the order of 0.17%. Furthermore if, for Class 0 instances, we used the known optima rather than the column generation-based lower bound, this gap would reduce to 0.1%. Finally, even though the number of generated patterns may appear large, the column generation procedure is very efficient, requiring 20 seconds CPU in the worst case (500 instances; times are negligible for the other instance types).

The average performance results of the different versions of the FFD, BFD and Diving upper bound heuristics are summarized in Table 3. Columns 1, 2, and 3 give the instance parameters. Columns 4 to 7 and 8 to 11 show the mean gap (in %) from the column generation-based lower bound for the four variants of the FFD BFD heuristics, respectively, corresponding to the four sorting strategies of Section 5.1. Columns 12-13 and 14-15 display the mean gap (%) from the column generation-based lower bound and the mean CPU time (seconds) of the diving heuristics with the rule (19) and (20), respectively, for selecting the non-integer patterns to be set to 1. Note that 1) no computing times are reported for the constructive heuristics because these were less than 0.01 seconds on average, and 2) the diving heuristic computing times do not include the time required to compute the continuous lower bound.

TYPE	BINS	ITEMS	FFD				BFD				DIVING 1		DIVING 2	
			1	2	3	4	1	2	3	4	GAP	TIME	GAP	TIME
0	3	25	12.95	12.95	3.83	3.83	12.95	12.95	3.62	3.62	1.17	0.00	1.35	0.00
		50	13.35	13.35	2.53	2.53	13.35	13.35	2.45	2.45	1.00	0.01	0.81	0.01
		100	12.24	12.24	1.69	1.69	12.24	12.24	1.65	1.65	0.49	0.02	0.48	0.02
		200	10.30	10.30	1.31	1.31	10.30	10.30	1.27	1.27	0.42	0.12	0.43	0.17
		500	8.99	8.99	1.09	1.09	8.99	8.99	1.08	1.08	0.23	1.54	0.26	1.97
	5	25	10.58	10.58	2.01	2.01	10.58	10.58	2.01	2.01	0.55	0.00	0.48	0.00
		50	11.68	11.68	1.86	1.86	11.68	11.68	1.85	1.85	0.44	0.00	0.47	0.00
		100	10.68	10.68	1.32	1.32	10.68	10.68	1.31	1.31	0.37	0.02	0.39	0.02
		200	10.86	10.86	0.86	0.86	10.86	10.86	0.85	0.85	0.21	0.10	0.24	0.13
		500	10.45	10.45	0.66	0.66	10.45	10.45	0.64	0.64	0.11	0.98	0.16	1.55
MEAN			**11.21**	**11.21**	**1.71**	**1.71**	**11.21**	**11.21**	**1.67**	**1.67**	**0.50**	**0.28**	**0.51**	**0.39**
1	3	25	14.76	17.11	5.43	8.98	14.10	16.75	5.16	8.81	0.89	0.00	1.11	0.00
		50	15.82	17.99	5.43	7.67	15.05	17.96	5.04	7.61	1.15	0.00	1.01	0.00
		100	14.61	16.68	4.73	7.46	13.98	16.58	3.92	7.45	1.05	0.02	1.16	0.02
		200	13.50	15.20	3.99	7.18	12.61	15.17	3.48	7.17	0.89	0.11	1.19	0.12
		500	11.95	13.70	3.25	6.61	11.13	13.72	2.72	6.60	0.66	1.22	0.80	1.56
	5	25	9.73	14.97	4.90	8.46	9.55	14.65	5.03	8.43	0.96	0.00	0.91	0.00
		50	11.01	16.06	5.54	7.57	10.69	15.85	5.23	7.53	0.40	0.00	0.41	0.00
		100	10.29	14.43	4.52	6.48	10.13	14.35	3.87	6.46	0.34	0.01	0.46	0.02
		200	10.35	14.97	3.78	6.34	10.06	14.89	3.26	6.33	0.27	0.07	0.42	0.10
		500	9.99	14.88	3.21	6.31	9.79	14.86	2.64	6.31	0.26	0.81	0.50	1.35
MEAN			**12.20**	**15.60**	**4.48**	**7.31**	**11.71**	**15.48**	**4.03**	**7.27**	**0.69**	**0.22**	**0.80**	**0.32**
2	3	25	10.51	10.28	3.88	4.66	10.21	10.25	3.38	4.51	0.88	0.00	0.87	0.00
		50	12.49	12.05	3.97	4.56	12.10	12.00	3.44	4.52	0.55	0.00	0.91	0.00
		100	10.93	10.88	3.02	3.54	10.46	10.84	2.61	3.53	0.74	0.02	0.76	0.02
		200	9.89	9.60	2.87	3.36	9.35	9.60	2.44	3.32	0.63	0.13	0.81	0.15
		500	8.71	8.85	2.25	3.30	8.09	8.86	1.87	3.29	0.57	1.47	0.61	1.98
	5	25	7.66	9.40	3.86	3.90	7.62	9.30	3.61	3.90	0.41	0.00	0.54	0.00
		50	7.88	9.65	4.09	3.34	7.72	9.57	3.47	3.34	0.29	0.00	0.46	0.00
		100	7.15	9.20	3.20	3.62	7.01	9.16	2.85	3.61	0.24	0.02	0.28	0.02
		200	7.18	9.35	2.66	3.27	7.04	9.30	2.24	3.26	0.18	0.09	0.28	0.12
		500	6.96	9.23	2.24	3.15	6.86	9.23	1.87	3.14	0.26	1.02	0.43	1.65
MEAN			**8.94**	**9.85**	**3.20**	**3.67**	**8.64**	**9.81**	**2.78**	**3.64**	**0.47**	**0.28**	**0.59**	**0.39**
OVERALL MEAN			**10.78**	**12.22**	**3.13**	**4.23**	**10.52**	**12.17**	**2.83**	**4.19**	**0.55**	**0.26**	**0.63**	**0.37**

Table 3: Comparison of upper bound heuristics for GBPP as gaps in % to the lower bound

CLASS	BINS	ITEMS	*R-SC*	BEST DIVING	GAP	OPT	Z_{SC}	GAP	OPT
0	3	25	1607.54	1623.33	0.98	10	1611.67	0.26	12
		50	3133.71	3153.33	0.63	3	3141.67	0.25	3
		100	6257.76	6281.33	0.38	6	6266.00	0.13	6
		200	12469.18	12510.33	0.33	3	12488.33	0.15	3
		500	31303.53	31363.00	0.19	2	31379.67	0.27	2
	5	25	1591.99	1598.33	0.40	10	1595.00	0.19	13
		50	3154.74	3165.00	0.33	6	3158.33	0.11	9
		100	6373.85	6391.67	0.28	3	6378.67	0.08	4
		200	12463.53	12483.67	0.16	4	12468.67	0.04	4
		500	31381.67	31412.67	0.10	1	31390.33	0.03	2
MEAN					**0.43**			**0.12**	
1	3	25	-1260.49	-1251.90	0.68	13	-1257.77	0.22	15
		50	-2449.51	-2432.00	0.72	4	-2444.27	0.21	4
		100	-4931.44	-4893.73	0.76	2	-4923.43	0.16	2
		200	-9932.75	-9856.60	0.77	3	-9924.57	0.08	3
		500	-25268.70	-25124.53	0.57	1	-25235.97	0.13	1
	5	25	-1269.84	-1263.83	0.47	11	-1267.07	0.22	13
		50	-2483.14	-2476.17	0.28	9	-2480.70	0.10	12
		100	-5196.42	-5185.33	0.21	6	-5194.53	0.04	10
		200	-10173.20	-10153.37	0.19	5	-10170.97	0.02	6
		500	-25601.17	-25545.83	0.22	1	-25595.13	0.02	4
MEAN					**0.55**			**0.13**	
2	3	25	-2117.11	-2104.17	0.61	11	-2113.23	0.18	11
		50	-3868.00	-3850.47	0.45	2	-3862.83	0.13	4
		100	-8024.54	-7976.30	0.60	5	-8019.83	0.06	7
		200	-16185.98	-16098.47	0.54	2	-16176.03	0.06	3
		500	-40206.50	-40024.53	0.45	2	-40155.10	0.13	1
	5	25	-1894.28	-1888.30	0.32	14	-1892.07	0.12	15
		50	-4089.92	-4084.07	0.14	8	-4087.20	0.07	7
		100	-8067.18	-8053.23	0.17	5	-8064.23	0.04	6
		200	-16019.83	-15999.03	0.13	3	-16015.87	0.02	4
		500	-40166.43	-40080.57	0.21	2	-40158.00	0.02	4
MEAN					**0.42**			**0.09**	
OVERALL MEAN					**0.46**			**0.11**	

Table 4: Best upper and lower bound performances

The results show that the best sorting rule for both FFD and BFD is the third one, corresponding to bins sorted in non-decreasing order of C_j/W_j and non-increasing values of volumes, and non-compulsory items in non-increasing p_i/w_i ratios and non-increasing values of their volumes. It is noteworthy that in terms of the overall mean gap, the two heuristics display almost the same performance, with a slight advantage for the BFD. It thus appears that, while for other packing problems, the classical Bin Packing in particular, the gap between BFD and FFD may be large, it is relatively small for a generalized version of the problems such as GBPP. Actually, an instance-by-instance analysis of the results shows that neither heuristic dominates the other. Then, given the negligible computational times, the best fast heuristic for GBPP is to take the minimum between the results of FFD and BFD with bins and items sorted as described above.

With a somewhat higher computational effort, better results are obtained by the diving heuristics. These yield solutions with average gaps of around 0.7% from the lower bound with a mean computational time less than 0.4 seconds once the lower bound is computed. Although the first Diving heuristic seems to offer a slightly better performance, analyzing the results instance by instance shows that also in this case there is no dominance between the two versions. Thus, the best results can be obtained by computing both and taking the minimum.

The set of results presented in Table 4 compares the best settings for the proposed heuristics. The column generation process is initialized with the solution corresponding to the minimum between the FFD and BFD heuristics with the best bin and item sorting rule indicated above (bins in non-decreasing order of C_j/W_j and non-increasing values of volumes, and non-compulsory items in non-increasing p_i/w_i ratios and non-increasing values of their volumes). The results of this procedures appear in Column *R-SC*. The two diving heuristics are then applied and their minimum yields the best solution in the Best Diving column. Column Z_{SC} displays the same type of results as in the previous tables (20 CPU seconds). For each heuristic, we report the mean gap (%) from the column-generation lower bound, as well as the number of instances solved to optimality.

The best results are still given by Z_{SC}. The best-diving heuristic achieves a very good performance, however. The average gap is about 0.50% in a very short computational time, while Z_{SC} yields an average gap of 0.17% but in a much longer time, reaching the 20 seconds mark for the largest instances. Moreover, the diving heuristic is able to prove the optimality of about 150 instances out of 900, with a distribution of optimal solution which does not seem to depend upon the parameters used to generate the instances.

6.2 Advanced analyzes

We first examine the variation in solution quality offered by the best heuristics proposed when the cardinality of I^C, the set of the compulsory items, varies.

We selected 12 large size instances (500 items) with a representative mix of characteristics in terms of item volumes, item profits, and bin types and built five instance sets with 0%, 25%, 50%, 75%, and 100% compulsory items, by randomly selecting items to be compulsory for cases 2 to 4. We then applied the BFD constructive heuristic, the Z_{SC} heuristic with 20 CPU seconds, and the diving heuristics. Table 5a displays the average gaps (%) over the 12 instances with respect to the column generation lower bound for the three heuristics, where the best result among the different versions was used for the BFD and diving heuristics. Table 5b displays the corresponding computational measures expressed in CPU seconds.

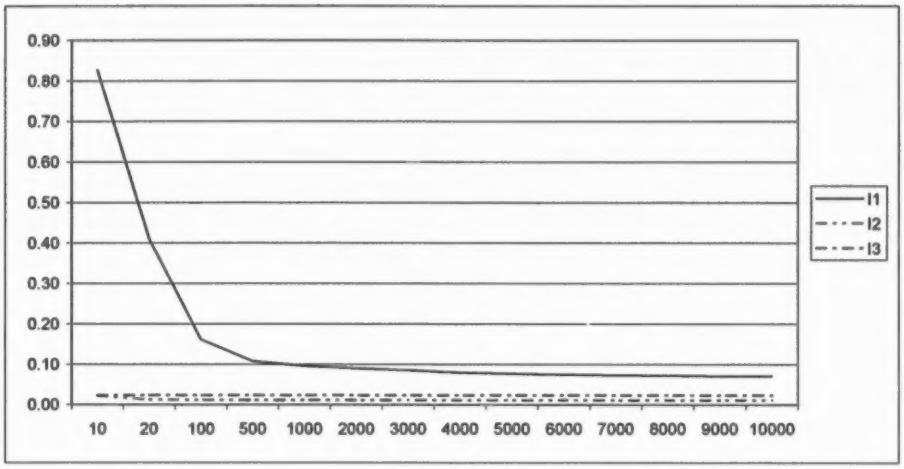
The results clearly show a trend. The instances requiring the most effort of all heuristics and for which these yield the poorest quality results are those where compulsory and non-compulsory items are more or less in equal numbers. All heuristics perform very well when one type of items dominates, performance degrading with the decrease in domination. The ranking observed previously is confirmed in this study: the constructive heuristic is extremely fast but yields results of poorer quality than the diving methods, which are outperformed by the Z_{SC} heuristic, i.e., solving the branch-&-bound of the set covering formulation for a limited time (20 CPU seconds) starting from the patterns generated by the column generation lower bound process.

It is worth noting that the performance of the Z_{SC} heuristic is very satisfactory even with respect to the most difficult instances. Moreover, it offers a very good global performance, with an average gap of 0.26%, an average computational effort of less than 12 seconds CPU, and 5 to 7 problems out of each group of 12 solved to optimality. Note that, we did not observe any correlation between this last number and the percentage of compulsory items, which seems to indicate that part of the gap for the the 50% instances comes from the optimality gap of the lower boud.

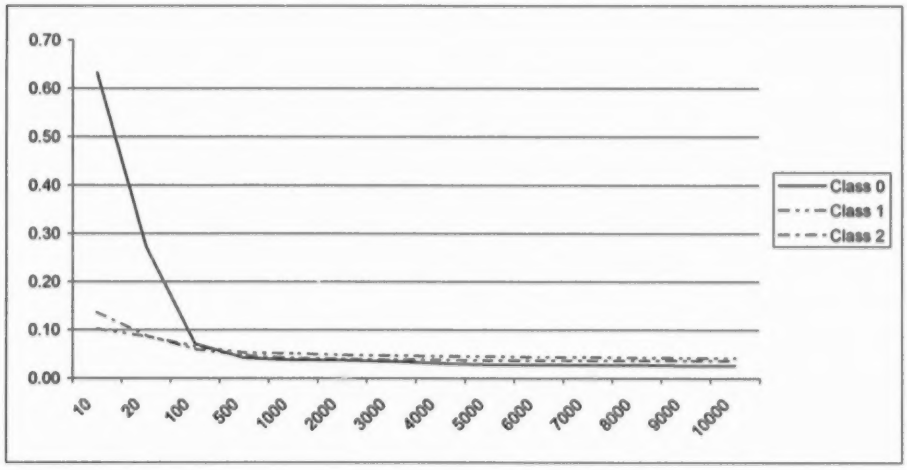
The experimental results discussed so far point to the Z_{SC} heuristic as the upper bound procedure offering the best performance in terms of solution quality. Because the procedure is based on solving a branch-&-bound algorithm for the MIP model SC, the heuristic may also serve as a good proxy to examine the behavior of exact methods in addressing the GBPP, as well as the impact of problem size on algorithmic efficiency.

A new set of experiments were thus performed by extending the time limit of the Z_{SC} heuristic to 10000 CPU seconds. All instances with up to 200 items were solved to optimality by XPress in at most 60 seconds, independently of the parameters used to generate the instance data. The behavior drastically changed when the number of items was increased to 500.

Figures 1a and 1b plot the evolution of the relative gap (in %) between the column generation-based lower bound and the best solution of the Z_{SC} heuristic as the computational effort, in CPU seconds, is increased. Instances are grouped by item size (I1, I2,



(a)



(b)

Figure 1: Computational effort versus quality gap of Z_{SC} heuristic for 500-item instances

I^C (%)	BEST bfd	Z_{SC}	BEST DIVING
0	2.15	0.07	0.63
25	3.90	0.13	1.14
50	21.40	0.71	7.16
75	2.77	0.15	0.63
100	0.86	0.27	0.21
MEAN	**6.22**	**0.26**	**1.95**

(a)

I^C (%)	BEST bfd	Z_{SC}	BEST DIVING
0	< 0.01	12.21	2.15
25	< 0.01	12.08	1.98
50	< 0.01	13.43	2.04
75	< 0.01	10.61	2.19
100	< 0.01	10.40	2.34
MEAN	< 0.01	**11.75**	**2.14**

(b)

Table 5: Performance of best heuristics with varying cardinality of I^C

and I3) in Figure 1a, while the item profit (Class 0, Class 1, and Class 2) are used to group instances in Figure 1b.

According to these results, it appears that instances with a wide variance in item sizes are more difficult to address with an un-tailored branch-&-bound than when the variation is smaller. The algorithm requires about 4000 seconds to reach a gap of less than 0.1% and then progresses at a slow rate. Part of this behavior may be explained by the fact, already mentioned previously, that the set of patterns generated while computing the lower bound for instances with very diverse item sizes may include a smaller number of the variables making up the optimal solution than for types of instances. This contributes to the interest of developing an exact method for the GBPP that takes advantage of the characteristics of this new class of problems and the quality of the proposed bounds.

Not surprisingly, the Z_{SC} is most challenged when all items are compulsory, i.e., belonging to Class 0, as it requires some 500 CPU seconds, on average to reach a gap under 0.1%. It is very encouraging, however, to witness this not-so-bad behavior on instances that are different (no item profits) from those the algorithms were developed for. This, combined to the excellent performance on the other instance classes, underlines the efficiency of the proposed algorithms and the interest to continue the research in this area.

7 Conclusions

We introduced the Generalized Bin Packing Problem, a new packing problem where, given a set of items characterized by volume and profit and a set of bins with given volumes and costs, one aims to select the subsets of profitable items and appropriate bins to optimize an objective function combining the cost of using the bins and the profit yielded by loading the selected items. The GBPP generalizes many other packing problems, including Bin Packing and Variable Sized Bin Packing, as well as Knapsack, Multiple Homogeneous and Heterogeneous Knapsack, and is relevant for many transportation and logistics planning problems.

We presented two formulations for the problem, based on item-to-bin assignment decisions and feasible loading patterns and set covering ideas, respectively. The latter proved more interesting in algorithmic terms, conducting to an efficient column generation-based lower bound method. We also presented first fit, best fit, and column generation-based upper bound procedures.

The paper also introduced new instance sets. The analysis of the results of extensive computational experiments showed that the proposed procedures are very efficient and the bounds are very tight.

Acknowledgments

While working on this project, the first author was the NSERC Industrial Research Chair on Logistics Management, ESG UQAM, and Adjunct Professor with the Department of Computer Science and Operations Research, Université de Montréal, and the Department of Economics and Business Administration, Molde University College, Norway. Funding for this project has been provided by the Politecnico di Torino and the Natural Sciences and Engineering Council of Canada (NSERC), through its Industrial Research Chair and Discovery Grants programs, and by the partners of the Chair, CN, Rona, Alimentation Couche-Tard and the Ministry of Transportation of Québec.

References

G. Wäscher, H. Haussner, H. Schumann, An Improved Typology of Cutting and Packing Problems, European Journal of Operational Research 183 (2007) 1109–30.

I. Correia, L. Gouveia, F. Saldanha-da-Gama, Solving the variable size bin packing problem with discretized formulations, Computers and Operations Research 35 (2008) 2103–13.

M. Monaci, Algorithms for packing and scheduling problems, Ph.D. thesis, Universitá degli Studi di Bologna, 2001.

T. G. Crainic, G. Perboli, W. Rei, R. Tadei, Efficient Heuristics for the Variable Size Bin Packing Problem with Fixed Costs, Tech. Rep., CIRRELT, 2010.

J.-F. Cordeau, G. Laporte, A. Mercier, A unified tabu search heuristic for vehicle routing problems with time windows, Journal of the Operational Research Society 52 (8) (2001) 928–36.

D. Pisinger, S. Ropke, A general heuristic for vehicle routing problems, Computers and Operations Research 34 (8) (2007) 2403–35.

S. Martello, P. Toth, Knapsack Problems - Algorithms and computer implementations, John Wiley & Sons, Chichester, UK, 1990.

S. P. Fekete, J. Schepers, New classes of lower bounds for bin packing problems, Mathematical Programming 91 (1) (2001) 11–31.

T. G. Crainic, G. Perboli, M. Pezzuto, R. Tadei, New bin packing fast lower bounds, Computers and Operations Research 34 (2007a) 3439–57.

T. G. Crainic, G. Perboli, M. Pezzuto, R. Tadei, Computing the Asymptotic Worst-case of Bin Packing Lower Bounds, European Journal of Operational Research 183 (2007b) 1295–303.

F. Vanderbeck, Computational study of a column generation algorithm for bin packing and cutting stock problems, Mathematical Programming 86 (1996) 565–94.

F. de la Vega, W. Lueker, Bin packing can be solved within $1 + \epsilon$ in linear time., Combinatorica 1 (1981) 349–55.

N. Karmarkar, R. M. Karp, An efficient approximation scheme for the one-dimensional bin packing problem, in: Proc. 23rd Annual Symp. Found. Comp. Sci. (FOCS 1982), 312–20, 1982.

P. Schwerin, G. Wäscher, The Bin-Packing Problem: A Problem Generator and Some Numerical Experiments with FFD Packing and MTP 4 (2006) 377–89.

J. Kang, S. Park, Algorithms for the variable sized bin packing problem, European Journal of Operational Research 147 (2003) 365–72.

S. S. Seiden, R. Van Stee, L. Epstein, New bounds for variable-sized online bin packing, SIAM Journal on Computing 32 (2) (2003) 455–69.

D. Pisinger, M. Sigurd, The two-dimensional bin packing problem with variable bin sizes and costs, Discrete Optimization 2 (2005) 154–67.

C. Alves, J. Valerio De Carvalho, Accelerating column generation for variable sized bin-packing problems, European Journal of Operational Research 183 (2007) 1333–52.

H. Dyckhoff, A Typology of Cutting and Packing Problems, European Journal of Operational Research 44 (1990) 145–59.

A. Atamturk, M. Savelsbergh, Integer-programming software systems, Annals of Operations Research 140 (2005) 67–124.

Dash Associates, XPRESS-MP User guide and reference manual (Release 2008a), Dash Associates, Northants, 2008.

P. C. Gilmore, R. E. Gomory, A linear programming approach to the cutting stock problem - part II, Operations Research 11 (1963) 863–88.

P. C. Gilmore, R. E. Gomory, A linear programming approach to the cutting-stock problem, Operations Research 9 (1961) 849–59.